# Multipath Optimized Link State Routing for Mobile ad hoc Networks

Jiazi Yi, Hassiba Asmaa Adnane, Sylvain David, Benoît Parrein

▶ **To cite this version:**

Jiazi Yi, Hassiba Asmaa Adnane, Sylvain David, Benoît Parrein. Multipath Optimized Link State Routing for Mobile ad hoc Networks. Ad Hoc Networks, 2011, 9 (1), pp.28-47. 10.1016/j.adhoc.2010.04.007 . hal-00521719

**HAL Id: hal-00521719**
**https://hal.science/hal-00521719**

Submitted on 28 Sep 2010

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Multipath Optimized Link State Routing for Mobile ad hoc Networks

Jiazi YI, Asmaa ADNANE, Sylvain DAVID, Benoît PARREIN

*Université de Nantes, Nantes Atlantique Universités*
*IRCCyN, CNRS UMR 6597, Polytech'Nantes, rue Christian Pauc - BP50609 44306 Nantes cedex 3 France*
*Email: {firstname.lastname}@univ-nantes.fr*

**Abstract**

Multipath routing protocols for Mobile Ad hoc NETwork (MANET) address the problem of scalability, security (confidentiality and integrity), lifetime of networks, instability of wireless transmissions, and their adaptation to applications.

Our protocol, called MP-OLSR (MultiPath OLSR), is a multipath routing protocol based on OLSR [1]. The *Multipath Dijkstra Algorithm* is proposed to obtain multiple paths. The algorithm gains great flexibility and extensibility by employing different link metrics and cost functions. In addition, *route recovery* and *loop detection* are implemented in MP-OLSR in order to improve quality of service regarding OLSR. The backward compatibility with OLSR based on IP source routing is also studied. Simulation based on Qualnet simulator is performed in different scenarios. A testbed is also set up to validate the protocol in real world. The results reveal that MP-OLSR is suitable for mobile, large and dense networks with large traffic, and could satisfy critical multimedia applications with high on time constraints.

*Key words:* MANET, OLSR, multipath routing, MP-OLSR, testbed, backward compatibility.

## 1. Introduction

Staying connected anywhere to a network is really the main objective of mobile technologies. Mobile Ad hoc NETwork (MANET) may provide a solution. With MANET, all nodes are routers and forward packets without any infrastructure. This kind of network is spontaneous, self-organized and self-maintained. In this context, routing the data is the big challenging task since many issues are covered: scalability, security, lifetime of network, wireless transmissions, increasing needs of applications.

Many routing protocols have been developed for ad hoc networks [2]. They can be classified according to different criteria. The most important is by the type of route discovery. It enables to separate the routing protocols into two categories: proactive and reactive. In reactive protocols, e.g. Dynamic Source Routing (DSR [3]) and Ad hoc On-demand Distance Vector routing (AODV [4]), the routing request is sent on-demand: if a node wants to communicate with another, then it broadcasts a route request and expects a response from the destination. Conversely, proactive protocols update their routing information continuously in order to have a permanent overview of the network topology (e.g. OLSR [1]).

Another criterion for ad hoc routing protocol classification is the number of routes computed between source and destination: multipath and single path routing protocols. Unlike its wired counterpart, the ad hoc network is more prone to both link and node failures due to expired node power or node mobility. As a result, the route used for routing might break down for different reasons. To increase the routing resilience against link or/and node failures, one solution is to route a message via multiple disjoint paths simultaneously. Thus, the destination node is still able to receive the message even if there is only one surviving routing path. This approach attempts to mainly address the problems of the scalability, mobility and link instability of the network. The multipath approach takes advantage from the large and dense networks.

Several multipath routing protocols were proposed for ad hoc networks [5]. The main objectives of multipath routing protocols are to provide reliable communication and to ensure load balancing as well as to improve quality of service (QoS) of ad hoc and mobile networks. Other goals of multipath routing protocols are to improve delay, to reduce overhead and to maximize network life time.

Multiple paths can be used as backup route or be employed simultaneously for parallel data transmission (like round robin). The multiple paths obtained can be grouped into three categories:

1. Disjoint: this group can be classified into node-disjoint and link-disjoint. In the node-disjoint multipath type, there are no shared nodes between the calculated paths that links source and destination. The link-disjoint multipath type may share some nodes, but all the links are different.
2. Inter-twisted: The inter-twisted multipath type may share one or more route links.
3. Hybrid paths: the combination of previous two kinds.

Of all the multipath types, the node-disjoint type is the most disjointed, as all the nodes/links of two routes are different i.e. the network resource is exclusive for the respective routes. Nevertheless, the pure disjoint approach is not always the optimal solution, especially for sparse networks and multi-criteria com-

puting. As we will see, our Multipath Dijkstra algorithm is more flexible when keeping all the solutions in the shortest paths algorithm.

In this paper, we started from the MultiPath Optimized Link State Routing protocol (MP-OLSR) presented in [6] which was thoroughly revisited and upgraded. Contributions are multiple. First, a major modification of Dijkstra algorithm allows for multiple paths both for sparse and dense topology. Two cost functions are used to generate node-disjoint or link-disjoint paths. Second, the OLSR proactive behavior is changed for an on-demand computation. MP-OLSR becomes a source routing protocol. Third, to support the frequent topology changes of the network, auxiliary functions, i.e. *route recovery* and *loop check*, are implemented. The contribution of these two functions is quantified in terms of quality of service parameters and compared with OLSR. Fourth, the backward and forward compatibility study with its single path version (OLSR) is proposed. The cooperation between the two protocols is expected here to facilitate the application and deployment of the new protocol. Simulations and real testbed demonstrate all the contributions.

The remainder of the paper is organized as follows. In section 2, related works on multipath routing protocols are summarized. In section 3, we introduce our protocol MP-OLSR and its auxiliary functionalities. Simulation and performance evaluation are presented in section 4. Section 5 presents the testbed and provides related test results. Compatibility between OLSR and MP-OLSR is studied in section 6. Finally, we conclude this paper.

## 2. Related works

In this section, we will first present the current situation of OLSR standardization, which includes both OLSR version 1 and OLSR version 2. Then some typical multiple path routing protocols for MANET are presented. And a related study based on testbed for MANET is introduced at the end.

### *2.1. OLSR version 1 and OLSR version 2*

OLSR, the most popular proactive routing protocol for ad hoc networks and OLSR version 1 (OLSRv1), has been standardized as an experimental RFC [1]. It is a link state protocol in which each node will send out *HELLO* and *TC* (Topology Control) messages periodically. It reduces the overhead of flooding link state information by requiring just MPR (Multi Point Relay) to forward the *TC* messages. A routing table is maintained to keep the next hop information to all the possible destination nodes.

OLSR version 2 (OLSRv2) has the same algorithm and ideas as OLSRv1. Being modular by design, OLSRv2 is made up from a number of generalized building blocks, standardized independently and applicable also for other MANET protocols. Currently, RFC 5148 - Jitter Considerations in Mobile Ad Hoc Networks [6], RFC 5444 - Generalized MANET Packet / Message Format [7] and RFC 5497 - Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs) [8] are published as RFCs, with the remaining constituent parts (MANET Neighborhood Discovery Protocol [9] and OLSRv2 [10]) being in the final phases of standardization. It has a more modular and extensible architecture, and is simpler and more efficient than OLSRv1. The multipath and its compatibility that we propose can also exist as additional modules in the OLSRv2 framework.

### *2.2. Multipath routing protocol for ad hoc networks*

Most of the proposed multipath protocols are based on the single path version of an existing routing protocol: AODV and AOMDV [11], DSR and SMR [12].

Most of these protocols are based on a reactive routing protocol (AODV [4] or DSR [3]). In fact, reactive multipath routing protocols improve network performances (load balancing, delay and energy efficiency), but they also have some disadvantages:

- Route request storm: multipath reactive routing protocols can generate a large number of route request messages. When the intermediate nodes have to process duplicate request messages, redundant overhead packets can be introduced in the networks [13].
- Inefficient route discovery: To find node-disjoint or link-disjoint paths, some multipath routing protocols prevent an intermediate node from sending a reply from its route cache [14]. Thus, a source node has to wait until a destination replies. Hence, the route discovery process of a multipath routing protocol takes longer compared to that of DSR or AODV protocols.

Compared to reactive routing, the proactive routing protocols need to send periodic control messages. Hence, several researchers consider proactive routing protocols as not suitable for ad hoc networks [5]. For a network with low mobility and network load, the reactive routing protocols generate fewer control messages. However, given a network with high mobility and large traffic, the cost of *route discovery* and *route maintenance* will rise significantly. On the other hand, the proactive protocols try to keep a routing table for all possible destinations and therefore provides a transmission delay shorter than reactive routing protocols [15]. Furthermore, because the proactive protocols try to maintain the information of the whole network by periodical control messages, they can discover multiple routes more efficiently without much extra cost.

Few studies were interested in multipath routing based on the OLSR protocol. Kun *& al.* [16] propose another version of multipath OLSR using IP source routing. Based on Dijkstra algorithm, the node calculates multiple paths to the destination. The calculated paths are strictly node-disjoint. The path is inserted in the IP header of the packet before sending. Based on these multiple paths, the paper introduces an algorithm of *load_assigned* to transmit data through the paths based on the congestion information of all the intermediate nodes on each path. The congestion information of one path is measured as the maximal size of the queue of the intermediate nodes (the queue size of a node is encapsulated in *HELLO* packets and advertised in *TC* messages). The algorithm of *load_assigned* selects two paths to transmit data according to their congestion information, and balances the load on the selected paths. In

[17], the authors also propose a similar algorithm to calculate node-disjoint multiple paths by removing used nodes from the topology information base. However, the mere source routing is problematic especially with topology changes which will be analyzed in the following section. In both studies, strict node disjoint routes are not always necessary and the suppression of nodes in multiple calls of Dijkstra algorithm could not work for sparse networks. The node-disjoint multiple paths are not suitable for partition or fusion group of nodes that can temporarily imply a single link for connection. Furthermore, the backward compatibility is not considered, which might be very important for the deployment of the new protocol.

In [18], the authors propose a multipath calculation based on the *shortest-widest path algorithm* for the protocol *QOLSR*, where *QOLSR* is an enhancement of the OLSR routing protocol to support multiple-metric routing criteria (bandwidth and delay). The proposed algorithm computes multiple loop-free and node-disjoint paths with a small correlation factor based on the delay and bandwidth metrics. The correlation factor is defined as the number of links connecting two disjointed paths. It is calculated to minimize interference between the multiple paths in order to achieve better QoS guarantees to applications and improve network resource utilization. However, the authors did not prove that the correlation factor can be correctly calculated. Indeed, OLSR nodes cannot have a global view of the network, but only links advertised in *HELLO* and *TC* messages (by default, the advertised link set in *TC* of the node is limited to the MPR selector set [1]). Moreover, this approach assumes a freshness of the measures (bandwidth, delay) which is difficult to obtain and maintain in practice. Some other metrics might be easier to obtain as mentioned in section 3.1.

In our work, we propose a new multipath Dijkstra algorithm, which provides node-disjoint or link-disjoint paths when necessary by adjusting distinct cost functions. Additional functionalities are used to adapt to the topology changes.

### *2.3. Testbed for ad hoc networks*

A high number of network protocols are only assessed through simulators due to implementation difficulties. This might induce two problems: firstly, with current simulation technology, it is not easy to simulate the exact real world scenario, especially the behavior in the physical layer model. In [19], the authors use intelligent ray tracing model to simulate a more realistic physical layer with a very high cost (3 days on a 50-node PC cluster to produce 120 GB of output data for just one scenario). The results show that there are differences between the commonly used physical layer model (free space or two ray ground) and the more realistic physical layer.

Secondly, some of the techniques and network parameters are easy to achieve in a simulator, but not in practice. For example, some of the protocols use extra information, such as delay and bandwidth as link metrics [18] to improve the performance of the network without mentioning how to obtain and maintain this kind of real-time information. This information might be easy to get in the simulator, but it is not very practical for a general usage.

Consequently, we believe it is important not only to test the protocol with the simulator, but also to validate it in a real testbed to ensure that it is practical and feasable with current technology.

In [20], the authors describe their experiments building a multi-hop wireless ad hoc network of eight nodes based on DSR protocol driving around a 700m by 300m site. The jitter introduced by the network is measured and a push-to-talk voice service is tested. For OLSR, one of the most sophisticated implementations is *OLSR daemon* (*olsrd*[21]) which is highly portable and scalable. It now runs on community mesh networks of up to 2000 nodes (*Athens wireless network* [22]) and 400 nodes (*FunkFeuer.at net* [23]). The team from Niigata University, Japan, implemented OLSRv2 in a testbed with 50 nodes on their campus [24] and concluded that address compression and link layer notification can improve the performance of OLSRv2.

The implementation of multipath routing is rare in the literature. In [25] and [26], the authors propose the multipath testbed based on multipath DSR in the following of [20]. And in [27], a multipath testbed is implemented based on multipath AODV. In [28], the authors set up a testbed in OMF framework based on Greedy Dominating Set algorithm [29] for gateway placements in mesh networks. The results obtained from these testbeds show that the multipath routing protocol can provide shorter average route recovery time and higher throughput. For our study, in addition to the simulation results, we have implemented MP-OLSR in a real testbed to validate our protocol, which will be introduced in the rest of the article.

## 3. Multipath OLSR - Functionalities

The MP-OLSR can be regarded as a kind of hybrid multipath routing protocol which combines the proactive and reactive features. It sends out *HELLO* and *TC* messages periodically to detect the network topology, just like OLSR. However, MP-OLSR does not always keep a routing table. It only computes the multiple routes when data packets need to be sent out.

The core functionality of MP-OLSR has two parts: *topology sensing* and *route computation*. The *topology sensing* is to make the nodes aware of the topology information of the network. This part benefits from *MPRs* like OLSR. The *route computation* uses the *Multipath Dijkstra Algorithm* [30] [15] to calculate the multipath based on the information obtained from the *topology sensing*. The source route (all the hops from the source to the destination) is saved in the header of the data packets.

The *topology sensing* and *route computation* make it possible to find multiple paths from source to destination. In the specification of the algorithm, the paths will be available and loop-free. However, in practice, the situation will be much more complicated due to the change of the topology and the instability of the wireless medium. So *route recovery* and *loop detection* are also proposed as auxiliary functionalities to improve the performance of the protocol. The *route recovery* can effectively reduce the packet loss, and the *loop detection* can

be used to avoid potential loops in the network as depicted in subsection 3.3 and 3.4.

In this section, we discuss both the core functionalities and auxiliary functionalities.

### 3.1. Topology Sensing

To get the topology information of the network, the nodes use *Topology sensing* which includes link sensing, neighbor detection and topology discovery, just like OLSR [1].

Link sensing populates the local link information base (*Link Set*). It is exclusively concerned with OLSR interface addresses and the ability to exchange packets between such OLSR interfaces. Neighbor detection populates the neighborhood information base (*Neighbor Set* and *2-hop Neighbor Set*) and concerns itself with nodes and node main addresses. Both link sensing and neighbor detection are based on the periodic exchange of *HELLO* messages. Topology Discovery generates the information base which concerns the nodes that are more than two hops away (*Topology Set*). It is based on the flooding of the *TC* messages (optimized by selecting the MPR set).

Through topology sensing, each node in the network can get sufficient information of the topology to enable routing. The link state protocol tries to keep the link information of the whole network as mentioned above. By default, the path quality is measured by the number of hops according to [1]. It can also be measured by other metrics such as BER (Bit Error Rate)[31] or the queue length. In our previous work [31], the BER metric showed better performance in certain scenarios, but the benefit is not obvious in various situations (such as urban areas). ETX metric [32] is also proposed as a MANET Internet Draft and is bound to become a standard. It has been extensively used in mesh networks around the world. In [33], the authors present a comparison between OLSR-RFC default hysteresis/hop count metric and OLSR-ETX metric in a mesh network testbed. However, their results reveal the ETX metric to be fundamentally flawed when estimating optimal routes in large dense mesh network and worse than the OLSR RFC standard. From the lower-layer point of view, the metrics still need to be further studied. And we also believe that the way the metrics are used for path selection can be service-dependent to improve the QoS.

What kind of link metric to use and how it can be used properly in MANET are still open topics [34]. In this paper, we follow the RFC of OLSR [1], which uses hop count as link metric. However, different metrics can be easily appended to *HELLO* or *TC* messages by using the extensible architecture of OLSRv2 [7].

### 3.2. Route Computation

In OLSR, routes are determined by nodes each time they receive a new topology control messages (TC or *HELLO*). The routes to all the possible destinations are saved in the routing table. For MP-OLSR, an on-demand scheme is used to avoid the heavy computation of multiple routes for every possible destination. First, the multipath computation hypotheses will be introduced prior to presenting the resulting algorithm.

#### 3.2.1. Hypotheses

The aim of the multipath algorithm is to build a set $\mathcal{K}$ of $N$ paths, with no loops, joining a source node (noted $s$) and a destination node (noted $d$).

An ad hoc network can be represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ where $\mathcal{V}$ is the set of vertices, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ the set of arcs and $c : \mathcal{V} \rightarrow \mathcal{R}^{*+}$ a strictly positive cost function. We assume the graph to be initially undirected i.e. $(v_1, v_2) \in \mathcal{E} \Rightarrow (v_2, v_1) \in \mathcal{E}$ and $c(v_1, v_2) = c(v_2, v_1)$ and loopless, i.e. no arcs join a node to itself. We also assume that no pair of vertices can be connected by more than one arc. Given an ordered pair of distinct vertices $(s, d)$ we can define a path between $s$ and $d$ as a sequence of vertices $(v_1, v_2, ..., v_m)$ so that $(v_q, v_{q+1}) \in \mathcal{E}$, $v_1 = s$ and $v_m = d$.

The above representation necessarily implies to define what the cost function $c$ refers to in an ad hoc context. The cost is incremental and the smaller the link, the better. Different metrics can be applied as mentioned in section 3.1.

#### 3.2.2. Multipath Dijkstra Algorithm

For a source node $s$ in the network, MP-OLSR will keep an *updated flag* for every possible node in the network to identify the validity of the routes to the corresponding node. Initially, for every node $i$, the $updatedFlag_i$ is set to *false*, which means the route to the corresponding destination does not exist or needs to be renewed. When there is a route request to a certain node $i$, the source node will first check the $updatedFlag_i$.

- If the $updatedFlag_i$ equals *false*, the node will perform Algorithm 1 to get the multiple paths to node $i$, save it into the multipath routing table, and renew the corresponding $updatedFlag_i$ to *true*.
- If the $udpdatedFlag_i$ equals *true*, the node will find a valid route to node $i$ in the multipath routing table.

Every time the node receives a new *TC* or *HELLO* message and results in the changes in the topology information base, all the $updatedFlag$s will be set to *false*.

The algorithm to obtain the $N$ paths from $s$ to $d$ is detailed in Algorithm 1.

The proposed algorithm is applied to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$, two vertices $(s, d) \in \mathcal{E}^2$ and a strictly positive integer $N$. It provides an $N$-uple $(P_1, P_2, ..., P_N)$ of $(s, d)$-paths extracted from $\mathcal{G}$. $Dijkstra(\mathcal{G}, n)$ is the standard Dijkstra algorithm which provides the source tree of the shortest paths from vertex $n$ in graph $\mathcal{G}$; $GetPath(SourceTree, n)$ is the function that extracts the shortest-path to $n$ from the source tree $SourceTree$; $Reverse(e)$ gives the opposite edge of $e$; $Head(e)$ provides the vertex edge to which $e$ points.

The incremental functions $f_p : \mathcal{R}^{*+} \rightarrow \mathcal{R}^{*+}$ and $f_e : \mathcal{R}^{*+} \rightarrow \mathcal{R}^{*+}$ are used at each step to get a disjoint path between $s$ and $d$. $f_p$ is used to increase the costs of the arcs that belong to the previous path $P_i$ (or the opposite arcs belonging to it). This will make future paths tend to use different arcs. $f_e$ is used to increase the costs of the arcs that lead to vertices of the previous path $P_i$. Therefore, there are three possible settings:

**Algorithm 1** Calculate $N$ routes in $\mathcal{G}$ from $s$ to $d$

```
MultiPathDijkstra(s, d, G, N)
c_1 ← c
G_1 ← G
for i ← 1 to N do
    SourceTree_i ← Dijkstra(G_i, s)
    P_i ← GetPath(SourceTree_i, d)
    for all arcs e in E do
        if e is in P_i OR Reverse(e) is in P_i then
            c_{i+1}(e) ← f_p(c_i(e))
        else if the vertex Head(e) is in P_i then
            c_{i+1}(e) ← f_e(c_i(e))
        else
            c_{i+1}(e) ← c_i(e)
        end if
    end for
    G_{i+1} ← (V, E, c_{i+1})
end for
return (P_1, P_2, ..., P_N)
```

- if $id = f_e < f_p$, paths tend to be arc-disjoint;
- if $id < f_e = f_p$, paths tend to be vertex-disjoint;
- if $id < f_e < f_p$, paths also tend to be vertex-disjoint, but when not possible they tend to be arc-disjoint.

where $id$ is the identity function.

By using the cost functions, we can expect to find diversity in the $N$ paths regarding the network topology. But contrary to providing strictly node-disjoint paths, the multiple paths generated by our algorithm do not need to be completely disjoint. The reason for this choice is that the number of disjoint paths is limited to the $(s, d)$ minimal cut (defined as the size of the smallest subset of edges one cannot avoid in order to connect $s$ to $d$). This minimal cut is often determined by the source and destination neighborhoods. For example, if $s$ only has 3 distinct neighbors, one cannot generate more than 3 disjoint paths from $s$ to $d$. As a consequence, this limitation of diversity may be local, the rest of the network being wide enough to provide far more than 3 disjoint paths. Another drawback of completely disjoint paths algorithms is that it may generate very long paths since every local "cutoff" can only be used once.

For example, in Figure 1, node $S$ is trying to get multiple paths to node $D$. For *MultiPath Dijkstra Algorithm*, we use the number of hops as link cost metric and set $f_p(c) = 3c$ and $f_e(c) = 2c$ (more penalty to the used links). Initially, the cost for all the links is set to 1. For the first step, the shortest path S→A→B→G→D will be found. Then the cost functions will be used to increase the cost of the related arcs:

- S→A, A→B, B→G and G→D will be changed from 1 to 3 by using $f_p$.
- S→C and F→G will be changed from 1 to 2 by using $f_e$.

Then for the next step, the second shortest path S→C→E→F→G→D will be found. If we use the algorithm proposed in [17], and we delete the intermediate node A, B and G after the first step, it is impossible to obtain the second path.

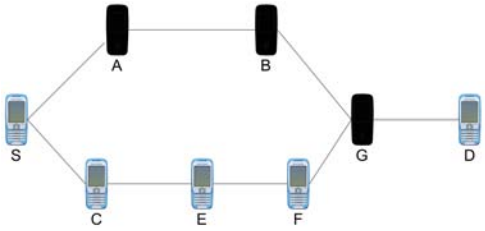


Figure 1: Multiple Dijkstra Algorithm in sparse case. The node disjoint path is non-desirable after node A, B and G are removed.

As illustrated above, another benefit of using cost functions is that we can get a different multiple path set (node-disjoint or link-disjoint) by choosing different cost functions according to our preference and the network requirements. The network topology in Figure 2 is presented as an example.

If we choose $f_p(c) = 3c$ and $f_e(c) = c$ (penalty is only applied to the used links), the paths we obtain are two link-disjoint paths: S→A→C→B→D and S→E→C→H→D.

If we choose $f_p(c) = 3c$ and $f_e(c) = 2c$ (penalty is also applied to used nodes), then the algorithm tends to search for node-disjoint paths. Then S→A→C→B→D and S→E→F→G→H→D will be found.

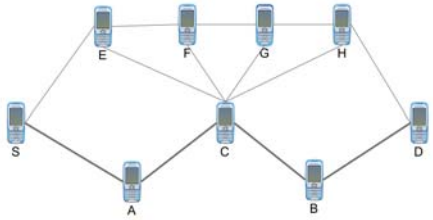


Figure 2: Obtaining different path sets by using different cost functions. Path S→A→C→B→D will first be chosen but the second one might be link disjoint or node disjoint (the bold lines) depending on the choice of cost functions.

### *3.3. Route Recovery*

By using the scheme of the *Topology Sensing*, we can obtain the topology information of the network with the exchange of *HELLO* and *TC* messages. All this information is saved in the topology information base of the local node: link set, neighbor set or topology set. Ideally, the topology information base can be consistent with the real topology of the network. However, in reality, it is hard to achieve, mainly because of the mobility of the ad hoc network.

Firstly, for the *HELLO* and *TC* messages, there are certain intervals during each message generation (2s for *HELLO* and 5s for *TC* by default [1]). During this period, the topology might change because of the movement of the nodes. Secondly, when the control messages (especially the *TC* messages) are being transmitted in the network, delay or collision might happen.

This will result in the control message being outdated or even lost.

Both of the two reasons mentioned above will result in the inconsistency between the real network topology and the node's topology information base. This means that when a node is computing the multiple paths based on the information base, it might use links that do not exist anymore, and cause the route failure.

Several techniques already exist in the literature to deal with the route failures in source routing. DSR handles route errors using route maintenance, mainly by sending RERR messages, which will increase the end-to-end delay significantly. In [35], the authors propose another method to avoid the effect of short term link deterioration by using opportunistic paths in mesh networks.

For MP-OLSR, we propose *Route Recovery* to overcome the disadvantage of the source routing. The principle is very simple: before an intermediate node tries to forward a packet to the next hop according to the source route, the node first checks whether the next hop in the source route is one of its neighbors (by checking the neighbor set). If so, the packet is forwarded normally. If not, then it is possible that the "next hop" is not available anymore. Then the node will recompute the route and forward the packet by using the new route.

In Figure 3 we present an example of route recovery. Node *S* is trying to send packets to *D*. The original multiple paths we have are S→A→B→D and S→C→E→G→D. However, node *G* moves out of the transmission range of node *E* and makes the second path unavailable. The source node *S* is not able to detect the link failure immediately (because of the delay and long interval of *TC* messages) and keeps sending the packets along the path, and all these packets are dropped during this period if only the source routing is used. With *Route Recovery*, when the packet arrives, node *E* will first check if node *G* is still one of its neighbors, before forwarding the packet according to the source route. If not, node *E* will recompute the route to node *D*, and obtain E→F→D. Then the following packets will be sent through the new path.

Because the *Route Recovery* just checks the topology information saved in the local node, it will not introduce much extra delay. And most importantly, it will effectively improve the packet delivery ratio of the network. In our simulation, the delivery ratio of the protocol with *Route Recovery* is 50% higher on average than the one without *Route Recovery*. In fact, the SR-MPOLSR [17] also has a very low delivery ratio like MP-OLSR without *route recovery* in our settings. This means that the mere source routing based on OLSR is not adapted.

### *3.4. Loop Detection*

Loop in the network is always an important issue in routing. It is important to mention the LLN (Link Layer Notification) before taking the problem of the loops of the protocol. LLN is an extended functionality defined in [1], and implemented in different OLSR or MP-OLSR simulations and implementations [30, 36]. If link layer information describing connectivity to neighboring nodes is available (i.e. loss of connectivity through

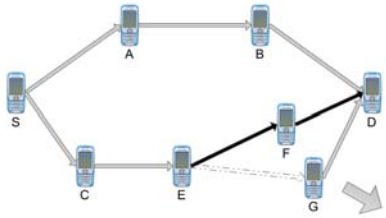


Figure 3: An example of route recovery. The movement of node G makes the link *E* to *G* unavailable. Then node F is chosen as next hop of node E by using route recovery

absence of a link layer acknowledgement), this information can be used in addition to the information from the *HELLO* message to maintain the neighbor information base and the MPR selector set. The routing protocol can act on the acknowledgement from LLN (mainly the loss of links), and remove the corresponding links from its information base. The results of the real OLSRv2 testbed [24] and our previous work [30] based on NS2 [37] simulation show that LLN is very important and effectively improves the packet delivery ratio of the OLSR and MP-OLSR protocol.

In theory, the paths generated by the Dijkstra algorithm in MP-OLSR are loop-free. However, in reality, the LLN and *Route Recovery* which are used to adapt to the topology changes make the loops possible in the network. With LLN, when a node tries to send a packet over a link but fails in the end, the link layer will send feedback to the routing protocol to notify it of the link loss. This kind of abrupt interruption will result in additional operations on the topology information base rather than just regular *HELLO* and *TC* messages. This means that other nodes cannot be aware of these changes immediately. So, LLN might cause some inconsistency of the topology information in different nodes. And with *Route Recovery*, which might change the path in intermediate nodes, loops can occur temporarily in the network.

In Figure 4 we give an example of how a loop is generated in the network. Node *A* is an intermediate node of a path. The packets with source route A→C arrive at node *A* and need to be forwarded to node *C*. Then node *C* moves out of the transmission range of node *A* and node *B*, and makes the links A→C, B→C no longer available.

When the new packets arrive at node *A*, the transmission to node *C* will be failed. Then in node *A*, the routing protocol will be acknowledged by LLN, and it will remove the link A→C from node *A*'s link set. For node *A*, although it can detect the link failure of A→C by LLN, it is hard to know the failure of B→C immediately. This is because link B→C can only be removed when the *NEIGHB_HOLD_TIME* (6 seconds by default [1]) expires. In the meantime, *Route Recovery* will be awaken. A new path A→B→C will be established and the following packets will be forwarded along the new path. Then the packets will be redirected to node *B*. The same operation will be performed in node *B*: LLN of the failure of B→C, and *Route Recovery*. Unfortunately, because node *B* cannot detect the link failure of A→C immediately, the new path obtained by *Route*

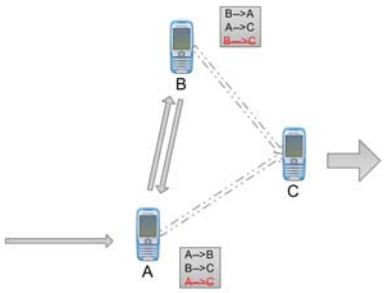


Figure 4: An example of loop in the network. The movement of node C results in inconsistency of the information bases in node A and B. One transient loop is formed between A and B.

*Recovery* is B→A→C. Thus the packet will be returned to node *A*, and from node *A* to *B* again, creating a loop. This is not a permanent loop, but a transient loop which will exist for several seconds and will disappear when the related link expires. However, this kind of temporary loops will block the links in the loop and congest the related transmission area.

In [38], the authors also address the looping issues in OLSRv2, and note that LLN will significantly increase the number of loops. Therefore, the authors introduce two types of loop detection techniques: LD-Mid (Mid-Loop Detection) and LD-Post (Post-Loop Detection). LD-Mid just compares the address of the next hop with the address of the previous hop, so it is only able to detect "two-way" loops between 2 nodes. LD-Post records all incoming packets that need to be forwarded and compares them with each new incoming packet to see if the same packet has been through this node before. So, it can detect loops that are farther away, by using more memory. When a loop is detected, the Packet Discard strategy is used to drop the packets that are unlikely to reach the destination but only increase the load of the network.

For MP-OLSR, we propose a simple method based on source routing that can effectively detect loops without causing extra cost of memory: after the *Route Recovery* is performed, a new path will be calculated from the current node to the destination. The algorithm will make use of the new path if there is no loop. Or else it will try to find another path according to the multipath algorithm. If there is no suitable path, the packet will be discarded.

For the example in Figure 4, node *A* will get a path A→B→C by *Route Recovery*. Then, when the packet arrives at node *B*, a new path B→A→C will be generated because of link breakage of B→C. Node *B* will compare the new one with the former source route A→B→C in the packet. We will find that the packet has already crossed node *A*, and so there might be a loop. So, the algorithm will try to find if there is any other possible path, or else the packet will be discarded.

Compared with LD-Post, which needs to keep a record of all the incoming packets, our loop detection mechanism could effectively detect the possible loops in the network without consuming extra memory space. By reducing the loops in the network, the network congestion can be reduced. Thus, the performance of the network can be improved, especially the end-to-end delay.

## 4. Simulation and Performance Evaluation

The simulations are performed to evaluate MP-OLSR which includes both the core functionality and the auxiliary functionality (*route recovery* and *loop detection*). The rest of the section is organized as follows. The simulation environment and assumption are first introduced in subsection 4.1. Then we compare the performances between OLSR and MP-OLSR in different scenarios. The difference between the reactive and proactive protocols is also analyzed. The performance evaluation will be done in a real testbed in section 5.

### *4.1. Environment and Assumption*

The Qualnet simulator 4.5.1 [39] is used for our simulation. It is the commercial version of GloMoSim and is widely used in academic research and industry. The MP-OLSR protocol is implemented on nOLSRv2 [36]. The scenarios include 81 nodes placed in an area of 1500m by 1500m to construct a mobile ad hoc network. The initial position of the nodes is uniformly distributed like a 9 ×9 grid. The nodes will move at certain speed according to the *Random Way Point Model (RWP)*. The maximum speed is 10m/s to simulate pedestrian and cycle applications.

For each simulation, the total simulation time is 100 seconds. A simple CBR (Constant Bit Rate) UDP application runs at several nodes to measure the performance of data transmission. Each CBR application corresponds to a source-destination flow which generates UDP packets of 512 bytes at the source, at different rates. The flow starts 15s after the simulation begins, to allow enough exchange of the routing messages. The data transmission lasts for 80s to obtain an average behavior of each flow.

The 802.11b radio is used and the data rate is set to 11Mbps. We use the two-ray ground pathloss model, the constant shadowing model with a shadowing mean of 4.0 dB, and the transmission power is set to 15dBm. With these settings, the transmission distance is about 270 meters in our simulations. We repeat each simulation 100 times and give the average results. Different random seeds are used to have different scenarios. Different seeds will generate different pseudo-random sequences used in simulation. This will affect the mobility according to the mobility model, back off timers, the interference pattern, etc. The detailed parameters are listed in Table 1 for the purpose of repeatability. Those parameters are widely used in WiFi devices and simulation studies.

The parameters for OLSRv2 and MP-OLSR are presented in Table 2. For the multipath routing, according to previous work [15], 2 to 4 paths could offer a desired performance with acceptable complexity. Given the node density in our simulation scenario, the algorithm tries to exploit three paths. The Round-Robin packet-distribution scheme is used for packet distribution.

To compare the performances of the protocols, the following metrics are used.

Table 1: Simulator Parameter Set

| Parameter | Values |
|---|---|
| Simulator | Qualnet 4.5.1 |
| Routing Protocol | OLSRv2 and MP-OLSR |
| Simulation area | 1500m × 1500m |
| Mobility | RWP, max speed 0-10m/s |
| Simulation Time | 100 seconds |
| Applications | CBR |
| Application Packet size | 512 bytes |
| Transmission Interval | 0.1 s |
| CBR start-end | 15s - 95s |
| Transport Protocol | UDP |
| Network Protocol | IPv4 |
| IP Fragmentation Unit | 2048 |
| Priority Input Queue Size | 50000 |
| MAC Protocol | IEEE 802.11 |
| MAC Propagation delay | 1uS |
| Short Packet Transmit Limit | 7 |
| Long Packet Transmit Limit | 4 |
| Rtx Threshold | 0 |
| Physical Layer Model | PHY 802.11b |
| Wireless Channel Frequency | 2.4 GHz |
| Propagation Limit | -111.0 dBm |
| Pathloss Model | Two Ray Ground |
| Shadowing Model | Constant |
| Shadowing Mean | 4.0 dB |
| Transmission Range | 270m |
| Temperature | 290K |
| Noise Factor | 10.0 |
| Receive Sensitivity | -83.0 |
| Transmission Power | 15.0dBm |
| Data Rate | 11Mbps |

- Packet Delivery Ratio: the ratio of the data packets successfully delivered at destination.
- Average end-to-end Delay: averaged over all surviving data packets from the sources to the destinations. This includes queuing delay and propagation delay.
- Average Time in FIFO Queue: average time spent by packets in the queue.
- Distribution of delay of received packets: this measurement can give an idea of the jitter effect.

### *4.2. Comparison between MP-OLSR and OLSR*

In this subsection, the performances of MP-OLSR and OLSR in different scenarios with different metrics are compared. The MP-OLSR used in this subsection is always with the *route recovery* and *loop detection* functionalities.

#### *4.2.1. Scenario with 81 nodes and 4 sources*

In Figure 5, the data delivery ratio of the two protocols is given. OLSR has a slightly better delivery ratio (about 3%) than MP-OLSR only at the speed of 1m/s (3.6km/h). This is because with more paths transmitting packets at the same time, there is a higher possibility of collision at the MAC layer. This inter-path interference can be eliminated by using multichannel techniques, which guarantee a different frequency band for each path [40]. In our case, there is only one channel used, so MP-OLSR has more packets dropped due to the collision at the MAC layer. However, as the speed of the nodes increases, the links become more unstable, and there are also more loops in the network. The delivery ratio of OLSR then decreases quickly and MP-OLSR outperforms OLSR.

Table 2: OLSR and MP-OLSR Parameters

| Parameter | Values |
|---|---|
| TC Interval | 5s |
| *HELLO* Interval | 2s |
| Refresh Timeout Interval | 2s |
| Neighbor Hold Time | 6s |
| Topology Hold Time | 15s |
| Duplicate Hold Time | 30s |
| Link Layer Notification | Yes |
| No. of path in MP-OLSR | 3 |
| MP-OLSR $f_e$ | $f_e(c) = 2c$ |
| MP-OLSR $f_p$ | $f_p(c) = 3c$ |

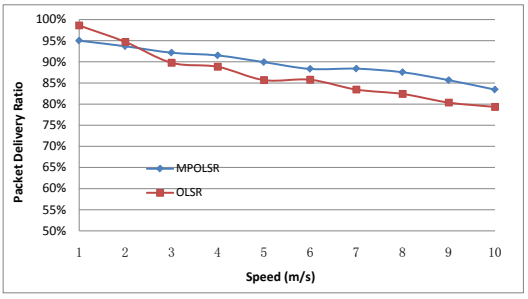


Figure 5: Delivery ratio of MP-OLSR and OLSR in a scenario of 81 nodes and 4 sources

Compared to the slight gain in the delivery ratio (about 5% at high speed), the multipath protocol performs much better on average end-to-end delay than the single path, as shown in Figure 6. The delay of OLSR is about 4 times more than MP-OLSR starting from 4m/s (14.4km/h). The end-to-end delay includes the *propagation delay* from the source to the destination and the *queue delay* in every relay nodes. The multipath protocol might have a longer propagation delay because some of the packets are forwarded through longer paths. However, what matters most is that it can effectively reduce the queue delay by distributing the packets to different paths rather than to a single one. In addition, the proposed loop detection mechanism can also reduce the unnecessary transmissions by avoiding the loops. As shown in Figure 7, the MP-OLSR has much shorter average time in the queue compared to OLSR.

In our simulations, the MP-OLSR also offers more stable delay in different simulations. Figures 8 and 9 show the distri-

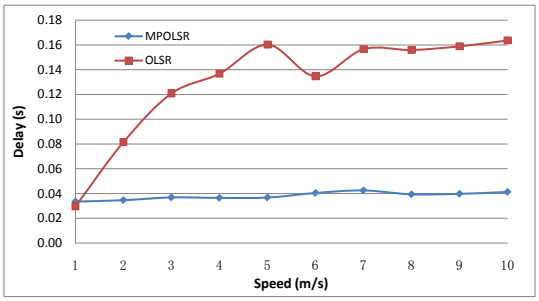


Figure 6: Average end-to-end delay of MP-OLSR and OLSR in a scenario fo 81 nodes and 4 sources

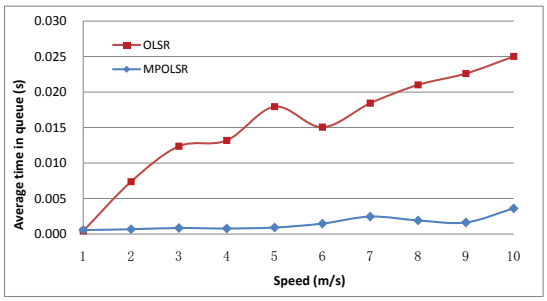


Figure 7: Average time in queue of MP-OLSR and OLSR in a scenario of 81 nodes, 4 sources

bution of end-to-end delay of all received packets in a scenario with medium mobility (0-5m/s). The distribution of the delay of OLSR spreads more widely compared to MP-OLSR. In this case, in the 2731 packets received by using OLSR, 1967 packets (82.96%) are received with delay less than 0.1s. For MP-OLSR, in the 2776 packets received, 2712 packets (97.69%) reach the destination in 0.1s. In fact, the standard deviation of the delay of OLSR is at least 10 times more than that of MP-OLSR, and even sometimes up to 100 times more in the high mobile scenarios.

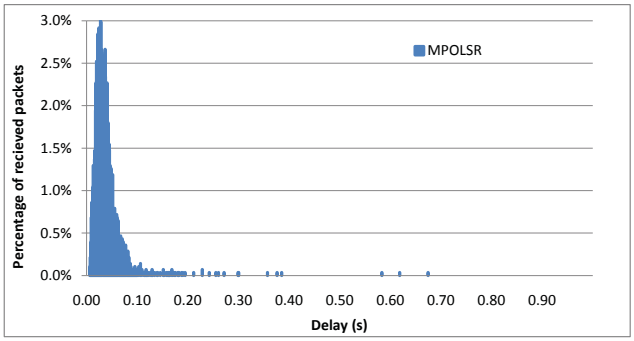


Figure 8: Distribution of delay of received packets of MP-OLSR in a scenario of 81 nodes, 4 sources

For the routing control message, because the MP-OLSR does not change the topology sensing mechanism of the OLSR protocol, the two protocols tend to have the same number of routing control messages generated. In our simulation scenarios, the number of *HELLO* messages and *TC* messages generated is almost the same.

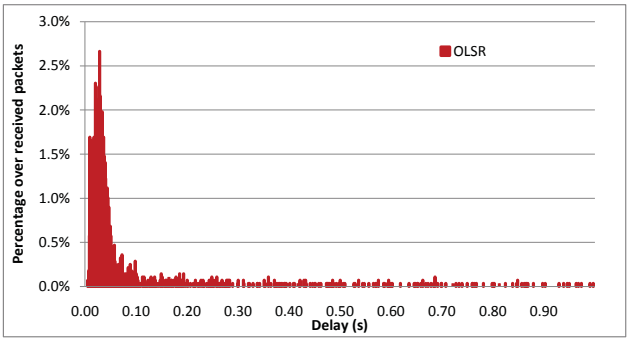


Figure 9: Distribution of delay of received packets of OLSR in a scenario of 81 nodes, 4 sources

### *4.2.2. Scenario with 81 nodes and 10 sources*

In this scenario, a higher load is applied to the network. There are 10 CBR sources transmitting the data packets to the destination, instead of 4 in the previous scenario in section 4.2.1.

In Figure 10, we present the delivery ratio of the two protocols. Compared to Figure 5 of the 4-source scenario, whose delivery ratio is always superior to 80%, the OLSR protocol is more unstable with the high load. With a speed superior to 6m/s (21.6km/h), it drops to about 65%. The MP-OLSR is more robust with high load, and stays at about 85% even with high mobility.

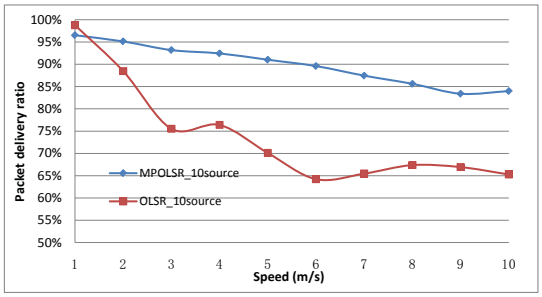


Figure 10: Delivery ratio of MP-OLSR and OLSR in a scenario of 81 nodes and 10 sources

Figure 11 is the average end-to-end delay. As we can see from the figure, the increase of the delay is much more significant than in Figure 6, which is more than 1 second at higher speed.

In this subsection, the protocols in different scenarios are simulated. The MP-OLSR and OLSR are compared in different scenarios. In addition to the scenarios presented in this section, we performed other scenarios like *duplex scenario* (two nodes send and receive packets from each other simultaneously, like VoIP application), *short-time scenario* (we have many more

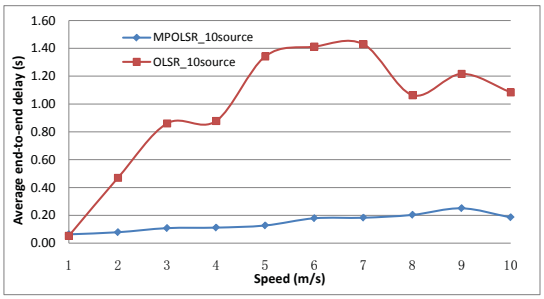


Figure 11: Average end-to-end delay of MP-OLSR and OLSR in a scenario of 81 nodes and 10 sources

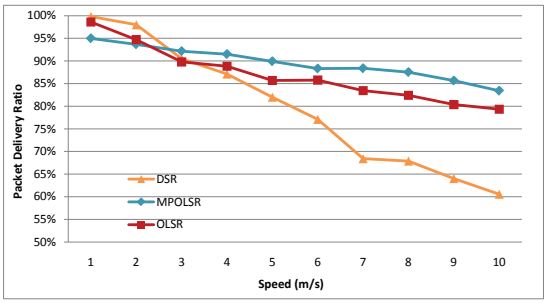


Figure 12: Delivery Ratio of DSR in a scenario of 81 nodes and 4 sources

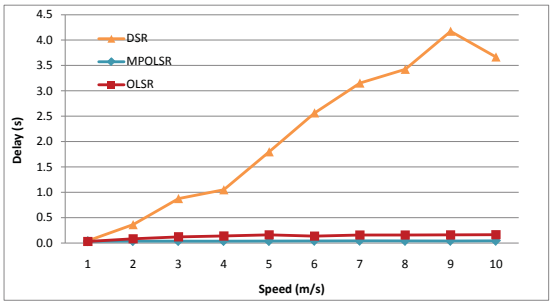


Figure 13: Average end-to-end elay of DSR in a scenario of 81 nodes and 4 sources

source-destination pairs, but very short transmission time, for an application like instant message sending). The results share the same trends and features as Section 4.2.1 and 4.2.2. The multipath protocol is more adapted to the mobile scenarios.

From the simulation results we obtained, we can conclude that, compared to OLSR, MP-OLSR has almost the same performance with low mobility and low network load. However, when the speed of the nodes or the network load increases, MP-OLSR has a better delivery ratio and a shorter delay than OLSR due to the ability to distribute the packets through different paths. In our settings, the speed is from 1m/s (3.6 km/h) to 10m/s (36 km/s). From the trends we observe that MP-OLSR offers more advantages than OLSR at even higher speed.

In real life, the topology changes can be caused by not only node movements, but also by the changes in the environment. Which are not taken into account in the simulation (e.g. the moving objects in the scenario or perturbation). All these will result in link failures. This phenomenon will be presented in the next section with the discussion of the testbed.

### *4.3. Comparison between proactive routing and reactive routing*

Because MP-OLSR is a hybrid routing protocol and uses source routing, we are also interested in the difference between proactive routing and reactive routing. The performance of DSR is also measured.

Figures 12 and 13 present the delivery ratio and delay of DSR respectively (compared to OLSR and MP-OLSR from section 4.2.1). The DSR has almost the same performance as the others protocols at low node speed (1m/s and 2m/s). However, when the mobility increases, the packet loss and delay of DSR increase significantly. In fact, the DSR uses source routing like MP-OLSR and also has a corresponding route recovery mechanism. But the reactive nature of DSR cannot adapt to frequent topology changes. Its route recovery mechanism is based on transmission of explicit RERR (route error) messages, which will increase rapidly as the node speed rises. On the other hand, the *route recovery* of MP-OLSR, which is based on link layer feedback and local network topology information base, does not need extra packet transmission in the network.

There are have been several multipath routing protocol proposed based on DSR, like SMR [12], which relies on the same reactive mechanism. The reactive feature makes those protocols hard to compare with the proactive ones in the scenarios with frequent topology changes. According to the simulation result of SMR [12], even if it can reduce the delay to 1/5 of DSR, it is still much higher than the proactive protocols.

## 5. Testbed

This section presents experimentation results of MP-OLSR based on the real testbed that we implemented. Two different scenarios are proposed in order to verify the MP-OLSR protocol and compare with OLSR. The following test was realized at the *Ecole Polytechnique of University of Nantes, Nantes, France*.

### *5.1. Hardware and Configuration*

We implemented MP-OLSR based on ASUS 901 EeePcs (figure 14) with the parameters shown in Table 3.

At the same time, we used the UFTP (UDP-based file transfer protocol) [41] application to test bi-directional exchanges. We used the following log files to analyze the results of each scenario:

- Wireshark log: log of packets captured by network interface in the network.
- MP-OLSR log: log of routing operations of MP-OLSR (link detection, route calculation with KDijkstra algorithm).

Figure 14: Some ASUS eeePCs 901 before deployment at the headquarters of the testbed

| Parameter | Value |
|---|---|
| CPU | Intel Atom N270 1.6GHz |
| Memory | DDR2 1024MB |
| Radio Frequency | 2.487 GHz |
| Rate | 54Mb/s with auto bit-rate |
| Physical Layer | 802.11g |
| Operating system | Linux (backtrack 3 live) |
| Kernel | Linux 2.6.21.5 |
| OLSR version | Olsrd 0.5.6r2 |
| Network protocol analyzer | Wireshark 0.91.6 |

Table 3: eeePC characteristics

- UFTP log: log of the process of the UFTP application.

Relying on this information, we measured the following parameters for each test:

- Test duration.
- Transfer duration: the duration of UFTP transmission, from the first to the last packet sent.
- Requested rate: the rate initialized by the user.
- Average rate: the number of data bytes actually delivered during the transfer.

### *5.2. Implementation of Multipath Routing*

The implementation of MP-OLSR is based on *Olsrd* [21]. The structure of the implementation is shown in Figure 15.

Like OLSR, after the reception of *HELLO* and *TC* messages, MP-OLSR updates the network topology information base. But for MP-OLSR, no further operation is performed because the routing table is not calculated at that moment. To send out user data, the TCP/IP stack sends a request to MP-OLSR to calculate a set of paths to the destination (for the first request or when the topology changes) or to return calculated routes (for the following requests).

Note that for the convenience of development and debugging, the MP-OLSR module currently exists as an application layer program in our setting. In the future, for efficiency sake and practical use, rather than in a testbed, it is better to put the MP-OLSR module in the Linux kernel as a kernel module to avoid frequent context switches after the test for the protocol is completed.

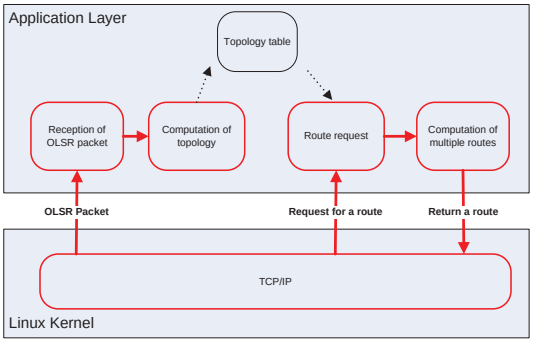


Figure 15: Implementation of MP-OLSR on Linux

### *5.3. Results*

In this subsection, two different scenarios are presented to compare the performance between multipath and single path protocols.

#### *5.3.1. Scenario 1, OLSR and MP-OLSR on 4 paths*

The first scenario presented includes 6 nodes. The location of the nodes is shown in Figure 16. The object is to test the multipath algorithm, so we try to find as many paths as possible in this simple scenario. The number of paths for MP-OLSR is set to 4. A node with an IP address 10.0.0.100 is chosen as source, and another node 10.0.0.98 as destination. The distance from the source node to the destination node is about 60 meters. There are different kinds of obstacles in this scenario: trees, buildings, and cars moving between the nodes, which will block certain links for a random period of time. *Iptable* rules are employed to block the direct transmission between source and destination to construct a multi-hop scenario.

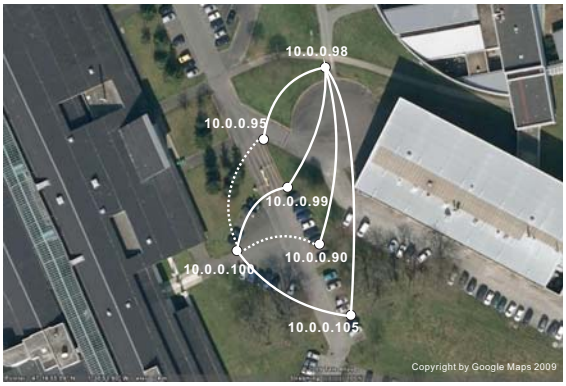


Figure 16: Network topology of the scenario 1: OLSR and MP-OLSR with 4 paths. 10.0.0.100 is the source and 10.0.0.98 is the destination

In the following tests, the data rate is set to 62 KBytes/s to transfer a file with 17.8 MBytes. Results are presented in Table 4. For MP-OLSR, the transmission was finished in 6 minutes 12 seconds. Nodes with IP addresses 10.0.0.90, 10.0.0.95, 10.0.0.99 and 10.0.0.105 are chosen as intermediate nodes to relay the packets. During the transmission, 9.9% of the packets were lost. For OLSR, only 10.0.0.90 and 10.0.0.95 are used to forward the data packets. The connection is lost after 5 minutes

17 seconds of transmission. For the packets sent out, 37.53% were lost.

| Protocol | MP-OLSR | OLSR |
|---|---|---|
| Duration of the transmission | 6m 12s | 5m17s (Connection lost) |
| Test duration | 7m50s | 6m26s |
| Size of the sent file | 17.8 MB | 17.8 MB |
| Requested rate | 62KB/s | 62KB/s |
| Average rate | 48.99 KB/s | 38.73 KB/s |
| Packets sent by 10.0.0.100 | 15002 | 9784 |
| Packets sent by 10.0.0.90 | 4503 | 5303 |
| Packets sent by 10.0.0.99 | 3715 | 0 (route not used) |
| Packets sent by 10.0.0.95 | 2084 | 2909 |
| Packets sent by 10.0.0.105 | 3726 | 0 (route not used) |
| Packets received by 10.0.0.98 | 13516 | 6112 |
| Rate of lost packets | 9.90% | 37.53% (Connection lost) |

Table 4: Results of scenario 1, OLSR and MP-OLSR with 4 paths, data rate = 62 KB/s

To compare the network performance of these two protocols, we also analyzed the log file from *Wireshark*. Figures 17 and 18 show the number of packets sent out to different nodes from the source (10.0.0.100) to the next hops in each tick (1 second per tick) for MP-OLSR and OLSR respectively.

As we can see from Figure 17, for a fixed source rate (10.0.0.100), the traffic load is distributed over 4 paths. The transmission is almost continuous even if some of the links are unavailable. If certain links break, the traffic will be assigned to other nodes (for example, from 290s to 300s and 380s to 420s).

Compared to MP-OLSR, the transmission with OLSR (Figure 18) is just through one path (so, the source rate is the same with the unique path and is not plotted here). The transmission is interrupted for a short period because the only path is unavailable. In this case, the node will try to find another route to the destination. But the data transmission will be stopped during this period (for example, 80s-90s, 115s-130s in figure 18). And if this kind of route switch takes too much time, the connection will be lost and result in the failure of the file transmission in the end.

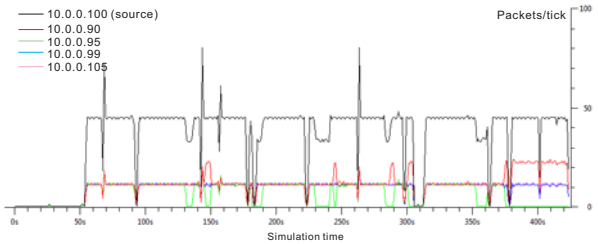


Figure 17: Wireshark trace of scenario 1 with MP-OLSR and rate=62KBytes/s

### *5.3.2. Scenario 2: OLSR and MP-OLSR routing on 3 paths*

In the second scenario, we compared OLSR and MP-OLSR with 3 paths which are three or four hops away. The goal is to test the protocol with longer paths in a complex scenario. The allocation of the nodes is shown in Figure 19. The distance from the source to the destination is about 200 m. There is a large

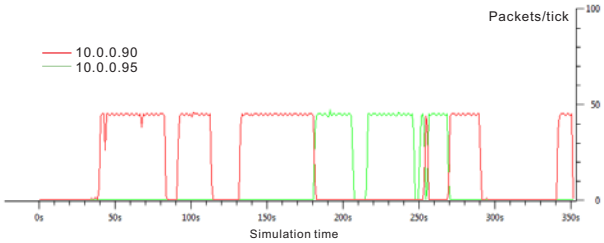


Figure 18: Wireshark trace of scenario 1 with OLSR and rate=62KBytes/s (Connection lost after 350 s)

building between them. To reach the destination, the packets have to travel around the building or go through the hall inside the building.

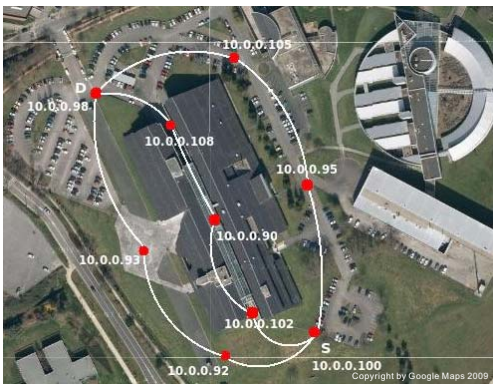


Figure 19: Network topology of scenario 2: OLSR and MP-OLSR with 3 paths, 10.0.0.100 is the source and 10.0.0.98 is the destination

Several links are unstable, mainly those in the parking-lot and inside the building (Figure 20). MP-OLSR only uses one or two paths most of the time. However, despite these frequent changes in the network topology, the transmission is successful with MP-OLSR.

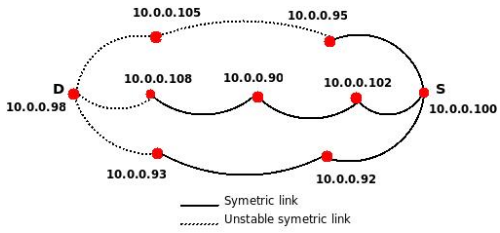


Figure 20: Quality of links in scenario 2: OLSR and MP-OLSR with 3 paths

With OLSR, the UFTP connection stopped after sending several packets. This is because compared to multiple paths, the unstable paths have more negative effects on the single path routing protocol. So the file transmission failed in the end because of time out. Test results are presented in Table 5.

In this subsection, experimentations are performed to show the efficiency and validity of the MP-OLSR routing protocol in real scenarios. UFTP is taken as application example. Mobility is not considered in the presented scenario, but the network topology still changes due to the failure of links.

| Protocol | MP-OLSR | OLSR |
|---|---|---|
| Duration of the transmission | 9m40s | 8m43s (Connection lost) |
| Test duration | 14m6s | 9m6s |
| Size of the sent file | 17.8 MB | 17.8 MB |
| Requested rate | 62KB/s | 62KB/s |
| Average rate | 31.42 KB/s | 34.85KB/s |
| Packets sent by 10.0.0.100 | 14528 | 12548 |
| Packets received by 10.0.0.98 | 9145 | 8544 |
| Rate of lost packets | 37.05% | 31.90% (Connection lost) |

Table 5: Results of scenario 2, OLSR and MP-OLSR routing with 3 paths

### *5.4. Discussion of simulation and testbed results*

If we compare the results from the simulator and the real testbed, we will find that the network performance in real testbed is not as good as that in the simulator, but with the same trend. This is reasonable because in the network simulation, we simulate the physical layer in a free space by using the ideal mathematical model (two-ray ground and shadowing). And in the real scenario, there are many more factors that will affect the final results: obstacles, radio reflection (much more complex than two-ray ground model), texture in the environment, radio interference (from other Wi-Fi devices, etc.), or even humidity and temperature. However, although it is hard to simulate the exact real scenario with current simulation technology, we can still have a relative evaluation between the protocols through the simulation results.

Because of the limitation of resources, it is very hard for us to perform the tests in the real scenario with a large number of nodes and mobility like in the simulation (for example, tens of nodes moving in an area of 1 $km^2$). As a compromise, to test the performance of the protocols with the real UFTP application in a more complex scenario, we set a semi-realistic testbed based on IP Network Emulation (IPNE) [42] interface, as shown in Figure 21. The IPNE implements a packet sniffer/injector. It sniffs the packets from the physical layer network, sends them through the Qualnet simulation, and injects them back to the physical network. The virtual Qualnet Network is transparent to the UFTP application.

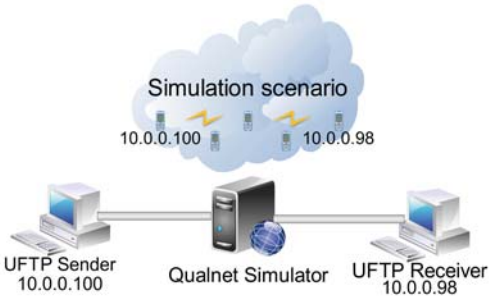


Figure 21: Semi-realistic IPNE testbed, with UFTP application

We launch the UFTP transmission at 100kbps, with the same under layer settings as in section 4. There are also 81 nodes in the network with medium mobility (maximum 5m/s). The Wireshark traces are shown in Figures 22 and 23 for OLSR and MP-OLSR respectively. The same thing happens with the tests in the real scenario, the file transmission using OLSR failed after a short period of time (170s). And for MP-OLSR, although it also suffers from the unstable paths during the same period, the file successfully reached the destination in the end. This result is coherent with the real testbed in section 5.3 .

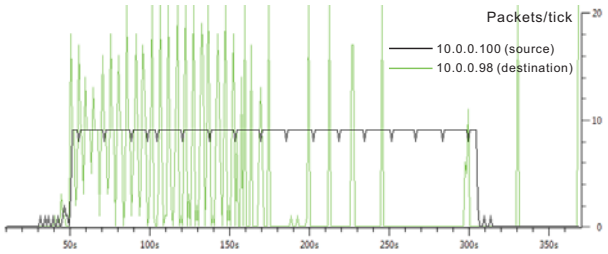


Figure 22: Wireshark trace of UFTP source and destination nodes, 81 nodes OLSR ad hoc network

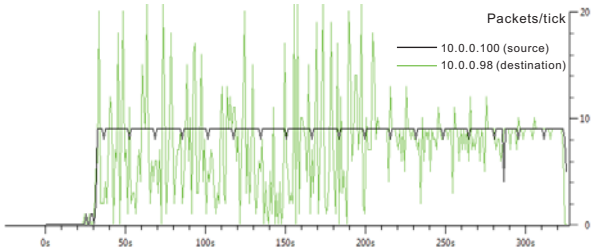


Figure 23: Wireshark trace of UFTP source and destination nodes, 81 nodes MP-OLSR ad hoc network

Based on the results obtained from the simulator and testbed, we can conclude that MP-OLSR could offer better performances than OLSR, especially in the harsh scenarios (high mobility in simulation and frequent link breakage in the testbed). This is mainly because the proactive-based multipath Dijkstra algorithm could find appropriate multiple paths and distribute the traffic into different paths by source routing. And the *route recovery* and *loop detection* could effectively avoid the disadvantage of source routing and the possible loops in the network.

## 6. Compatibility between MP-OLSR and OLSR

As presented in the related work, most of the multipath routing protocols proposed are based on single path version of an existing routing protocol: AODV and AOMDV [11], DSR and SMR [12]. However, the backward compatibility of those protocols with its single path version is not considered. In fact, because the reactive-based multipath routing requires extra operations during the route discovery to gather enough information for the multipath construction, the compatibility is not easy to achieve.

Given OLSR, the standardized protocol and the distributed/heterogeneity property of ad hoc networks, it will be interesting to study the compatibility between MP-OLSR and OLSR which will facilitate the application and deployment of the multipath routing.

### 6.1. The Problem of Compatibility

As presented in the introduction, MP-OLSR is based on the OLSR protocol. In fact, the two protocols follow the same steps for the detection of neighborhood and network topology, but they are different in routing the data packets. In OLSR, the source node calculates the shortest path to the destination and sends the packets to the next hop. The intermediate OLSR nodes forward the packets according to their routing table. In MP-OLSR, the source node calculates different paths, and specifies one path in each packet (source routing) before sending it to the next hop. And the intermediate MP-OLSR nodes will forward the packet according to the source routing initialized by the source node.

The study of backward compatibility makes the deployment of the new protocol much easier because it can make use of the network that already exists. Moreover, it allows to return to the single-path version if necessary (basically with no mobility and low traffic). It is important to point out that we are studying the mutual compatibility between OLSR and MP-OLSR. In the following, we show that each protocol can use the nodes of either protocol to perform routing, with respect to QoS parameters.

To ensure the compatibility between the two protocols OLSR and MP-OLSR, we propose an implementation of MP-OLSR protocol based on *IP-source routing* [43].

The IP source routing allows to partially or completely specify the path in the data packets. This option is mostly used by network administrators to test the routes. The IP protocol supports two forms of source routing:

1. Strict source routing: the exact route of the packet is specified by the sender.
2. Loose source routing: the sender gives one or more hops that the packet must go through. This means that before reaching its final destination, the packet should go through the following IP addresses as intermediate destinations. These intermediate destinations are responsible for forwarding it to the next destination.

The IP source routing option implicitly ensures the compatibility between OLSR and MP-OLSR. Indeed, when OLSR nodes receive an MP-OLSR packet (with source route), they will forward it directly according to the IP source routing. On the other hand, when an MP-OLSR node receives a packet generated by an OLSR node (without source route), it will recompute the path as the packet comes from its application layer and attach the source route to the packet. So this packet can also take advantage of the *loop detection* in all the following MP-OLSR intermediate nodes.

However, the IP source routing option accepts only a maximum of 9 addresses (in total, 11 nodes including source and destination nodes = 10 hops) because of the limitation of the length of the IP head. Therefore, when the route contains more than 10 hops (very large network), other solutions must be proposed. To solve this problem, we propose two possible solutions:

- The first is by using the *loose source routing*: the source node just specifies 10 "key" hops that the packet needs to travel through to reach the destination and allows each intermediate node to choose a route to the next hop. This solution can guarantee the source routing as defined by the source node. But it requires the *loose source routing* support from the IP layer and the MP-OLSR protocol to maintain a routing table just like OLSR.
- The second solution is that if the path found by MP-OLSR is more than 10 hops, it will just forward packets to the next hop, instead of using the source routing. The next hop will decide the rest of the route, no matter whether it is an MP-OLSR node or an OLSR node. This solution is easy to implement but does not guarantee the multiple paths as defined by the source node. In the following simulation, results are based on this solution.

### 6.2. Simulation results

In this section, we present routing performances when OLSR and MP-OLSR protocols cooperate in the same network in order to check the backward compatibility.

The following results are obtained with Qualnet simulator, and scenarios of 81 mobile nodes using a strict IP source routing. Nodes were uniformly placed initially on a square grid of 1500m×1500m. Table 1 summarizes the simulation settings.

#### 6.2.1. Scenario 1: network with MP-OLSR source nodes

In the first scenario, the simulation results are obtained for a network of 4 MP-OLSR source nodes. We change the density of the OLSR nodes. We start by studying a network of 4 MP-OLSR sources and all the rest 77 hosts are OLSR nodes (denoted *4mpolsr_77olsr*), then we replace OLSR nodes by MP-OLSR nodes, and for each scenario we note the number of MP-OLSR and OLSR nodes in the network (*20mpolsr_61olsr*, *40mpolsr_41olsr*, *60mpolsr_21olsr*). Finally, we give the results for a network of only MP-OLSR nodes.

Figure 24 shows the delivery ratio when the OLSR nodes are involved in the routing by carrying the packet generated by the MP-OLSR source nodes. In general, the OLSR nodes have no problem in forwarding the source routing packets generated by MP-OLSR nodes. However, the OLSR intermediate nodes cannot have the same performance with MP-OLSR nodes. This is mainly because OLSR cannot perform *route recovery* and *loop detection* for the packets. So, in the scenarios where the density of OLSR nodes is too high, it is better for MP-OLSR nodes to just forward the packet to the next hop without appending the source route. On the other hand, OLSR nodes do not significantly affect the average end-to-end delay of the MP-OLSR protocol (Figure 25).

#### 6.2.2. Scenario 2: network with OLSR source nodes

In the second scenario, we simulate the case in which the sources are OLSR nodes. Here, we have 4 OLSR source nodes with 77 MP-OLSR nodes, and we compare this scenario with that of all OLSR nodes. In Figures 26 and 27, we present the delivery ratio and average end-to-end delay respectively. As we can see in these figures, OLSR nodes have no problems in sending packets to MP-OLSR nodes. Furthermore, with the help of

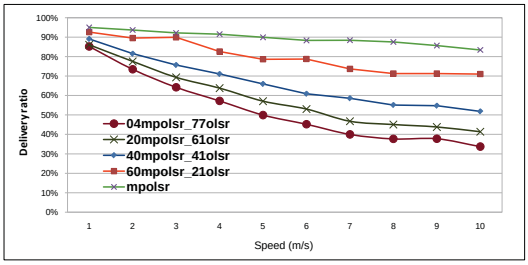


Figure 24: Ratio of delivered packets for a network of 81 OLSR and MP-OLSR mobile nodes, with MP-OLSR source nodes

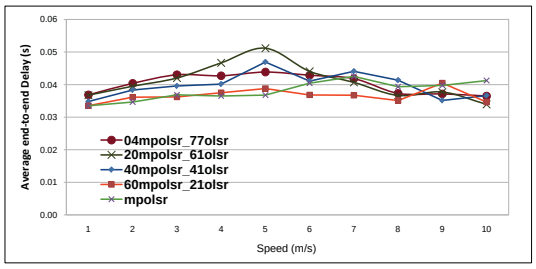


Figure 25: Average end-to-end delay for a network of 81 OLSR and MP-OLSR mobile nodes, with MP-OLSR source nodes

the *loop detection* and the multipath feature of MP-OLSR, we can increase the packet delivery ratio and reduce the end-to-end delay of the network.

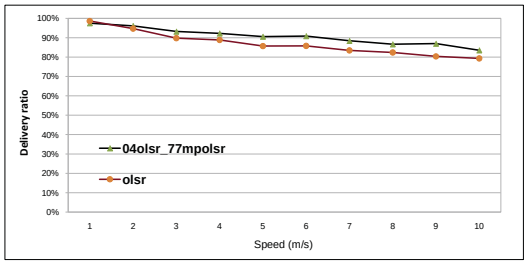


Figure 26: Ratio of delivered packets for a network of 81 OLSR and MP-OLSR mobile nodes, with OLSR source nodes

In conclusion, the MP-OLSR and OLSR can cooperate within in the same network. This feature makes the deployment of the MP-OLSR protocol much easier because it can make use of the existing OLSR network. Because MP-OLSR nodes can perform multipath routing and *loop detection*, it can improve the performance of the network. For OLSR nodes, the route failure might increase when using source routing, because they do not have *route recovery*. Therefore, when the density of OLSR nodes is very high, it is better for MP-OLSR to forward the packets without source route.


Figure 27: Average end-to-end delay for a network of 81 OLSR and MP-OLSR mobile nodes, with OLSR source nodes

## 7. Conclusion and Future Works

In this paper, the MultiPath Optimized Link State Routing (MP-OLSR) protocol is proposed. The extension of the single path version includes a major modification of the Dijkstra algorithm (two cost functions are now used to produce multiple disjoint or non-disjoint paths), auxiliary functions, i.e. *route recovery* and *loop detection* to guarantee quality of service and a possible backward compatibility based on IP source routing. The MP-OLSR can effectively improve the performance of the network (especially in the scenarios with high mobility and heavy network load) and also be compatible with OLSR. Simulations and real testbed demonstrate our contributions.

The advantages of a link state multipath approach are clearly exemplified. Classical issues in MANET are covered: scalability, lifetime of the network (by reducing the number of forwarded packets per node) and non reliable wireless transmissions. From the security point of view, we can argue that the multipath approach can increase confidentiality. By a spatial diversity, the classical Man-In-the-Middle (MiM) attack is quite ineffective assuming a large cooperation between applicants. For integrity purpose, a redundant coding can be integrated to the routing protocol in order to ensure an higher delivery rate [44].

The last point is the increasing needs of applications. The best benefit of MP-OLSR for QoS occurs in end-to-end delay and jitter that is precisely required for critical multimedia services. Routing decision based on different types of scalable streams (especially video streams) can be further exploited, combined with the study on the metrics of link quality to fulfil the QoS. This constitutes the subject of future work.

## Acknowledgements

This work is supported by the French program RNRT (Réseau National de Recherche en Télécommunications) under the project SEREADMO (Sécurisation des Réseaux ad hoc par Tansformée Mojette, ANR-05-RNRT-028-01). We also wish to thank the reviewers for their valuable comments and suggestions that helped improve the quality and presentation of this paper.

## References


[1] T. Clausen, P. Jacquet, IETF Request for Comments: 3626, Optimized Link State Routing Protocol OLSR (October 2003).

[2] M. Abolhasan, T. Wysocki, E. Dutkiewicz, A review of routing protocols for mobile ad hoc networks, Ad Hoc Networks 2 (1) (2004) 1 – 22.

[3] D. B. Johnson, Y. Hu, D. A. Maltz, IETF Request for Comments: 4728, The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4 (February 2007).

[4] C. Perkins, E. Royer, Ad-hoc on-demand distance vector routing, in: Second IEEE Workshop on Mobile Computing Systems and Applications, 1999, pp. 90–100.

[5] M. Tarique, K. E. Tepe, S. Adibi, S. Erfani, Survey of multipath routing protocols for mobile ad hoc networks, in: Journal of Network and Computer Applications, Vol. 32, 2009, pp. 1125–1143.

[6] T. Clausen, C. Dearlove, B. Adamson, IETF Request for Comments: 5148, Jitter Considerations in Mobile Ad Hoc Networks (February 2008).

[7] T. Clausen, C. Dearlove, J. Dean, C. Adjih, IETF Request for Comments: 5444, Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format (February 2009).

[8] T. Clausen, C. Dearlove, IETF Request for Comments: 5497, Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs) (March 2009).

[9] T. Clausen, C. Dearlove, J. Dean, IETF Internet Draft, MANET Neighborhood Discovery Protocol (NHDP), draft-ietf-manet-nhdp-10 (July 2009).

[10] T. Clausen, C. Dearlove, P. Jacquet, IETF Internet Draft, The Optimized Link State Routing Protocol version 2, draft-ietf-manet-olsrv2-10 (September 2009).

[11] M. K. Marina, S. R. Das, On-demand multi path distance vector routing in ad hoc networks, in: Proceedings of the Ninth International Conference on Network Protocols, IEEE Computer Society, Washington, DC, USA, 2001, pp. 14–23.

[12] S. Lee, M. Gerla, Split multipath routing with maximally disjoint paths in ad hoc networks, Helsinki, Finland, 2001, pp. 3201–3205.

[13] Z. Yao, J. J. Jiang, P. Fan, Z. Cao, V. Li, A neighbor table based multipath routing in ad hoc networks, in: 57th IEEE semi annual Vehicular Technology Conference, Vol. 3, 2003, pp. 1739–43.

[14] Z. Ye, S. V. Krishnamurthy, S. K. Tripathi, A framework for reliable routing in mobile ad hoc networks, in: IEEE INFOCOM, San Francisco, CA, USA, 2003, pp. 270–280.

[15] E. Cizeron, S. Hamma, A multiple description coding strategy for multipath in mobile ad hoc networks, in: International Conference on the Latest Advances in Networks (ICLAN), Paris, France, 2007.

[16] M. Kun, Y. Jingdong, R. Zhi, The research and simulation of multipath olsr for mobile ad hoc network, in: International Symposium on Communications and Information Technologies (ISCIT), 2005, pp. 540–543.

[17] X. Zhou, Y. Lu, B. Xi, A novel routing protocol for ad hoc sensor networks using multiple disjoint paths, in: 2nd International Conference on Broadband Networks, Boston, MA, USA, 2005.

[18] H. Badis, K. A. Agha, Qolsr multi-path routing for mobile ad hoc networks based on multiple metrics: bandwidth and delay, in: IEEE Vehicular Technology Conference, Los Angeles, CA, USA, 2004, pp. 2181–2184.

[19] I. Stepanov, K. Rothermel, On the impact of a more realistic physical layer on manet simulations results, Ad Hoc Networks 6 (1) (2008) 61 – 78.

[20] D. Maltz, J. Broch, D. Johnson, Lessons from a full-scale multihop wireless ad hoc network testbed, IEEE Personal Communications 8 (2001) 8–15.

[21] Olsrd, an adhoc wireless mesh routing daemon, http://www.olsr.org/.

[22] Athens wireless network, http://wind.awmn.net/?page=nodes.

[23] Funkfeuer.at, http://map.funkfeuer.at/.

[24] Y. Owada, T. Maeno, H. Imai, K. Mase, Olsrv2 implementation and performance evaluation with link layer feedback, in: Proceedings of the 2007 international conference on Wireless communications and mobile computing, Honolulu, Hawaii, USA, 2007.

[25] Y. Zhai, O. Yang, W. Wang, Y. Shu, Implementing multipath source routing in a wireless ad hoc network testbed, IEEE Pacific Rim Conference on Communications, Computers and signal Processing (2005) 292– 295.

[26] S. Mao, S. Lin, S. Panwar, Y. Wang, A multipath video streaming testbed for ad hoc networks, IEEE Vehicular Technology Conference (2003) 2961–2965.

[27] K. Taniyama, T. Morii, S. Koizumi, K. Noguchi, Experimental evaluation of an on-demand multipath routing protocol for video transmission in mobile ad hoc networks, Journal of Zhejiang University SCIENCE A (2006) 145–150.

[28] Omf, the testbed control and management framework, http://omf.mytestbed.net.

[29] J. W. Tsai, T. Moors, Minimum interference multipath routing using multiple gateways in mesh networks, in: IEEE Conference on Mobile ad-hoc and Sensor Systems, Atlanta, Georgia, 2008.

[30] J. Yi, E. Cizeron, S. Hamma, B. Parrein, Simulation and performance analysis of MP-OLSR for mobile ad hoc networks, in: IEEE WCNC: Wireless Communications and Networking Conference, Las Vegas, USA, 2008.

[31] A.-M. Poussard, W. Hamidouche, R. Vauzelle, Y. Pousset, B. Parrein, Realistic SISO and MIMO Physical Layer implemented in two Routing Protocols for Vehicular Ad hoc Network, in: IEEE ITST, Lille, France, 2009.

[32] H. Rogge, E. Baccelli, A. Kaplan, MANET Internet-Draft: Packet Sequence Number based ETX Metric for Mobile Ad Hoc Networks, http://www.ietf.org/id/draft-funkfeuer-manet-olsrv2-etx-00.txt (December 2009).

[33] D. Johnson, G. Hancke, Comparison of two routing metrics in olsr on a grid based mesh network, Ad Hoc Networks 7 (2009) 374–387.

[34] C. Dearlove, T. Clausen, P. Jacquet, Manet internet-draft: Link metrics for olsrv2, http://tools.ietf.org/html/draft-dearlove-olsrv2-metrics-04 (July 2009).

[35] J. W. Tsai, T. Moors, Opportunistic multipath routing in wireless mesh networks, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 22 (2009) 71–85.

[36] nOLSRv2, Niigata olsrv2 implementation, Niigata university, Japan.

[37] Network simulator 2, http://www.isi.edu/nsnam/ns/.

[38] L. Speakman, Y. Owada, K. Mase, An analysis of loop formation in OLSRv2 in ad-hoc networks and limiting its negative impact, in: IEEE International CQR Workshop, Naples, Florida, USA, 2008.

[39] Qualnet simulator, http://www.scalable-networks.com/products.

[40] H. Gharavi, Multichannel mobile ad hoc links for multimedia communications, Proceedings of the IEEE Vol. 96,Issue:1, 77–96.

[41] Udp based ftp with multicast, http://www.tcnj.edu/ bush/uftp.html.

[42] Scalable Network technologies, IP Network Emulation Interface.

[43] IETF Request for Comments: 791, IP: Internet Protocol (September 1981).

[44] N. Normand, B. Parrein, I. Svalbe, A. Kingston, Erasure coding with the fnite radon transform, in: IEEE Wireless Communications and Networking Conference, to appear, Sydney, Australia, 2010.